IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Jeff EDER

Serial No.: 09/940,450

Filed: August 29, 2001

For: An automated method of and system for identifying measuring and enhancing categories of value for a value chain

Group Art Unit: 3628

Examiner: C. Graham

Brief on Appeal

Commissioner of Patents and Trademarks

Washington, D.C. 20321

Sir or Madam:

The Table of Contents is on page 2 of this paper.

Table of Contents

Real party in interest

Asset Reliance, Inc. (dba Asset Trust, Inc.)

*

Related appeals

An appeal for U.S. Patent Application 10/012,374 filed December 12, 2001 may be affected by or have a bearing on this appeal. An appeal for U.S. Patent Application 10/329,172 filed December 23, 2002 may be affected by or have a bearing on this appeal. An appeal for U.S. Patent Application 09/761,671 filed January 18, 2001 may be affected or have a bearing on this appeal. An Appeal for U.S. Patent Application 09/764,068 filed on January 19, 2001 may be affected by or have a bearing on this appeal. An Appeal for U.S. Patent Application 09/688,983 filed on October 17, 2000 may be affected by or have a bearing on this appeal.

Status of Claims

Claims 34 – 52, 62 – 64, 68 – 70, 90, 91 and 134 are rejected and are the subject of this appeal. Claims 62, 68, 90, 91 and134 are amended. Claims 1 – 33, 53 – 61, 65 – 67, 71 – 89, and 92 – 133 were previously cancelled without prejudice. Claims 135 - 167 are new.

Status of Amendments

An Amendment/Reply after a Non-Final Rejection was submitted on September 3, 2006.

Summary of Claimed Subject Matter

One embodiment of an automated method of and system for identifying measuring and enhancing categories of value for a value chain according to the present invention is best depicted in Figures 1 – 10 of the specification for the instant application. Figure 1 gives an overview of the major processing steps which include integrating data from a plurality of database management systems for use in analysis, analyzing the data as required to develop a model of value chain performance by element and category of value, identify and analyze value improvements and produce reports.

One embodiment of an automated method of and system for identifying measuring and enhancing categories of value for a value chain is exemplified in independent claim 36 where a computer readable media causes a computer to implement a method that integrates data from a plurality of management systems using xml and a common schema. More specifically, data from the database management systems associated with a plurality of enterprise transaction systems are prepared for use in processing by integrating, converting and storing the data in accordance with a common schema as described in FIG. 1, reference number 200, FIG. 5A reference numbers 201 - 204, 207 – 209 and 211 FIG. 5B reference numbers 221 – 222, 225 – 226, 209 and 211, FIG. 5C reference numbers 241 – 242, 245 – 246, 209 and 211, FIG. 5D reference numbers 261 – 262, 265, 267 - 269, 209 and 211, FIG. 5E reference numbers 277 - 282, FIG. 5F reference numbers 292 - 297,and line 1, page 14 through line 35, page 43 of the specification.

A second embodiment of an automated method of and system for identifying measuring and enhancing categories of value for a value chain is exemplified in independent claim 44 where a method integrates data from a plurality of management systems using xml and a common schema in order to support organization processing. More specifically, data from the database management systems associated with a plurality of enterprise transaction systems are prepared for use in processing by integrating, converting and storing the data in accordance with a common schema as described in FIG. 1, reference number 200, FIG. 5A reference numbers 201 - 204, 207 – 209 and 211 FIG. 5B reference numbers 221 – 222, 225 – 226, 209 and 211, FIG. 5C reference numbers 241 – 242, 245 – 246, 209 and 211, FIG. 5D reference numbers 261 – 262, 265, 267 - 269, 209 and 211, FIG. 5E reference numbers 277 - 282, FIG. 5F reference numbers 292 - 297,and line 1, page 14 through line 35, page 43 of the specification.

A third embodiment of an automated method of and system for identifying measuring and enhancing categories of value for a value chain is exemplified in independent claim 62 where a computer readable media causes a computer to integrate data from a plurality of management systems for use in analysis in accordance with a common schema and analyze the data in order to create a plurality of tools for organization management. More specifically, data from the database management systems associated with a plurality of enterprise transaction systems are prepared for use in processing by integrating, converting and storing the data in accordance with a common schema as described in FIG. 1, reference number 200, FIG. 5A reference numbers 201 - 204, 207 – 209 and 211 FIG. 5B reference numbers 221 – 222, 225 – 226, 209 and 211, FIG. 5C reference numbers 241 – 242, 245 – 246, 209 and 211, FIG. 5D reference numbers 261 – 262, 265, 267, 269, 209 and 211, FIG. 5E reference numbers 277 - 282, FIG. 5F reference numbers 292 - 297,and line 1, page 14 through line 35, page 43 of the specification. The integrated data are then analyzed using a plurality of bots in order to develop a market value model for a value chain that uses analytical models, including network models, to identify a tangible impact of each element of value on each category of value and each component of value in accordance with the procedure detailed in FIG. 1, reference numbers 300 and 400,

FIG. 6A reference number 302 - 310, FIG. 6B reference numbers 321, 323 and 326 - 329, FIG. 6C reference numbers 341 - 343 and 345 – 350 FIG. 7 reference numbers 404, 404 and 409 – 415 and line 1, page 44 through line 18, page 65 of the specification. The value chain model is then used to develop management reports as described in FIG. 8 reference numbers 504 – 507 and line 20, page 65 through line 18, page 69 of the specification. The previously calculated information is then used to support the automated trading of organization equity as described in FIG. 8 reference numbers 509 – 512 and line 20 page 69 and line 18, page 70 of the specification. The value chain model is then used for simulation and optimization using the method described in FIG. 9 reference number 603 - 605 and 610 and lines 20, page 71 through line 17, page 73 of the specification. The results of the analyses include lists of changes that will optimize one or more aspects of organization financial performance. The results are reported using the method described in FIG. 9 reference number 611 and 612 and lines 20, page 73 through line 30, page 73 of the specification.

Grounds of rejection to be reviewed on appeal

Issue 1 - Whether claims 34 - 43 are patentable under 35 USC 103 over U.S. Patent 6,332,163 (hereinafter, Bowman-Amuah) in view of U.S. Patent 6,301,584 (hereinafter, Ranger)?

Issue 2 - Whether claims 44 - 52 are patentable under 35 USC 103 over Bowman-Amuah in view of Ranger?

Issue 3 - Whether claims 62 – 64, 68 – 70, 90, 91 and 134 are patentable under 35 USC 103 over Bowman-Amuah in view of Ranger?

The Argument

For each ground of rejection which Appellant contests herein which applies to more than one claim, such additional claims, to the extent separately identified and argued below, do not stand and fall together.

Issue 1 - Whether claims 34 – 43 are patentable under 35 USC 103 over Bowman-Amuah (US Patent 6,073,115) in view of Ranger (U.S. Patent 6,909,708)?

The claims are patentable because the cited combination of documents used to support the rejection of claims 34 – 43 fails to establish a prima facie case of obviousness. Reasons the cited combination fails to establish a prima facie case of obviousness include:

1. the cited combination of documents teach away from the proposed combination;
2. the cited combination of documents fails to make the invention as a whole obvious,
3. the cited combination fails to meet any of the criteria for establishing a prima facie case of obviousness, and
4. the cited combination requires a change in the principles governing the operation of one or more of the methods disclosed in the documents.

The first reason that claims 34 – 43 are patentable is that the cited combination of documents teaches away from their own combination. MPEP § 2145 X.D.2 provides that: "it is improper to combine references where the references teach away from their combination." The Bowman-Amuah and Ranger documents teach away from the proposed theoretical combination in a number of ways, including:

1. Incompatible data management. Ranger teaches a method for retrieving data in a response to a query that gathers information dynamically from one or more data sources which may be located at different servers and have incompatible formats (see page 20, Evidence Appendix, Ranger, abstract). Bowman-Amuah teaches a system that incorporates replication and synchronization services (see page 29, Evidence Appendix, Bowman-Amuah C49, L45 – C50, L55). A replicated database often consolidates data from heterogeneous data sources, thus shielding the user from the processes required to locate, access and query the data (see page 29, Evidence Appendix, Bowman-Amuah C50, L45 – 50). It clearly would be improper to combine an invention that teaches and relies on obtaining data using a time consuming procedure in response to a query with an invention that teaches and relies on replication and synchronization to eliminate the need for a time consuming procedure for obtaining data in response to a query.
2. Incompatible processing speeds. Bowman-Amuah teaches a fixed format stream-based communication system (see page 24, Evidence Appendix, Bowman-Amuah abstract). As described in the specification and shown in FIG. 20 of the Bowman Amuah specification (see pages 25 & 26, Evidence Appendix Bowman Amuah FIG. 20 and C3, L 6 -7) a stream based communication system involves the real time transmission of data. By way of contrast, the Ranger invention uses a time consuming process to retrieve data on a one at a time basis. In particular, the Ranger method invokes agents to gather data, post the results of their data gathering on to a common "blackboard" before integrating the data into one or more entities (see page 22, Evidence Appendix, Ranger, C18, L33 - 42). It clearly would be improper to combine an invention that teaches and

relies on obtaining data using a time consuming procedure on a one item at a time basis with an invention that teaches and relies on a real time data stream.

3. Incompatible focus. It is a specific goal of Bowman-Amuah to couple the access to data with the completion of business functions (see page 30, Evidence Appendix, Bowman-Amuah C284, L 9 – 11). By way of contrast, Ranger teaches a stand alone method for the aggregation of data. In particular, the Ranger method invokes agents to gather data, post the results of their data gathering on to a common "blackboard" before integrating the data into one or more entities (see page 22, Evidence Appendix, Ranger, C18, L33 - 42). It clearly would be improper to combine an invention that teaches the coupling of data collection with the completion of business functions with an invention that teaches the stand alone collection of data.
4. Overlapping functions. Bowman-Amuah teaches a fixed format stream-based communication system that incorporates the ability to efficiently retrieve data (see page 24, Evidence Appendix, Bowman-Amuah abstract). By way of contrast, the Ranger invention uses a time consuming process to retrieve data on a one at a time basis. In particular, the Ranger method invokes agents to gather data, post the results of their data gathering on to a common "blackboard" before integrating the data into one or more entities (see page 22, Evidence Appendix, Ranger, C18, L33 - 42). It clearly would be improper to combine an invention that teaches and relies on obtaining data using a time consuming procedure on a one item at a time basis with an invention that already has the capability to efficiently retrieve data.

The second reason claims 34 – 43 are patentable is that the cited combination of documents fails to make the invention as a whole obvious as required by MPEP § 2141.02 which states that: "in determining the difference between the prior art and the claims, the question under 35 U.S.C. 103 is not whether the differences themselves would have been obvious but whether the claimed invention as a whole would have been obvious." As noted previously, the obviousness rejections are based on a combination of Bowman-Amuah and Ranger. Each of the documents:

1. teach away from the claimed method in a number of ways, and
2. teach away from the proposed combination as described previously.

Taken together the cited combination of documents fails to make the invention as a whole obvious. The cited combination also fails to make a single aspect of the claimed invention obvious. These failures provide additional evidence that the claimed invention for producing, concrete, tangible and useful results is new, novel and non-obvious.

The third reason claims 34 - 43 are patentable is that the cited combination fails to meet any of the criteria required for establishing a prima facie case of obviousness. MPEP 2142 provides that: in order to establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation to modify the reference or combine the reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The 24 July 2006 Office Action fails to meet all three criteria for establishing a prima facie case of obviousness as detailed below:

1. The cited combination of documents fails to meet the first criteria for establishing a prima facie case of obviousness for claims 34 - 43 because it does not provide any evidence that there was any suggestion, teaching or motivation (including scientific reasoning) in the prior art to modify or combine the teachings of

Bowman-Amuah and/or Ranger. In fact the opposite is true as there is an incentive not to complete the theoretical combination of Ranger and Bowman-Amuah. It is well established that "teachings of references can be combined only if there is some suggestion or incentive to do so" quoting ACS Hosp. Sys., Inc. v Montefiore Hosp., 732 F.2d 1572, 1577 221 U.S.PQ 929,933 (Fed. Cir. 1984). Reasons for not completing the proposed theoretical combination include

a. Bowman-Amuah already has the ability to retrieve data (see page 24, Evidence Appendix, Bowman-Amuah abstract) and the Examiner has been unable to explain why the combination was suggested in the first place;
b. Bowman-Amuah teaches that xml is going to be displaced by SMIL (see page 28, Evidence Appendix, Bowman-Amuah C42, L5 - 25) which provides a motivation not to complete the proposed combination; and
c. Bowman-Amuah teaches that xml is only suitable for use in displaying web pages (see pages 27 and 28, Evidence Appendix, Bowman Amuah C40, L60 – C41, L6) which provides another motivation not to complete the combination.

2. The cited combination also fails to meet the second criteria for establishing a prima facie case of obviousness for claims 34 - 43 because it does not cite a combination of documents that has a reasonable expectation of success. There are several reasons why the cited combination of documents (Bowman-Amuah and Ranger) does not have a reasonable expectation of success. These reasons include:
   a. the two documents teach incompatible methods for data management;
   b. the proposed combination would destroy the ability of the Bowman-Amuah invention to support real-time, streaming communication systems, and
   c. the Examiner has been unable to explain how the combination would be completed. It is well established that "particular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed" (In re Kotzab, 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000)). In spite of this well know requirement, the Office Action has not described how the teachings of these references would be combined.
3. The cited combination fails to meet the third criteria because it does not teach or suggest one or more of the claim limitations for every rejected claim.

The fourth reason that claims 34 - 43 are patentable is that the proposed combination of documents would change one or more of the principles of operation of the Bowman-Amuah and Ranger methods. MPEP 2143.01 provides that when "the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims prima facie obvious. In re Ratti, 270 F.2d 810, 123 USPQ 349 (CCPA 1959)". As noted previously, the obviousness rejections are based on a combination of Bowman-Amuah and Ranger. Some of the changes in the operating principles of the cited documents that would be required to make the combination function are discussed below.

1. Elimination of replication and synchronization services. Bowman-Amuah teaches a system that incorporates replication and synchronization services (see page 29, Evidence Appendix, Bowman-Amuah C49, L45 – C50, L55). A replicated database often consolidates data from heterogeneous data sources, thus shielding the user from the processes required to locate, access and query the data (see page 29, Evidence Appendix, Bowman-Amuah C50, L45 – 50). The Examiner has proposed using Bowman-Amuah in combination with Ranger to among other things render obvious an

invention for the stand alone integration of data in accordance with a common schema. There would be no need to combine Bowman-Amuah with Ranger unless the replication and synchronization services of Bowman-Amuah were removed. Modifying Bowman-Amuah to eliminate replication and synchronization services would require a change in principle in the operation of the Bowman-Amuah invention. As a result, the teachings of the cited combination of documents are not sufficient to render the claims prima facie obvious.

2. Change from integrated data access and business analysis to stand alone data access. One of the specific goals of the invention described in the Bowman-Amuah document is to couple the access to data with the completion of business functions (see page 30, Evidence Appendix, Bowman-Amuah C284, L 9 – 11). The Examiner has proposed using Bowman-Amuah in combination with Ranger to among other things render obvious an invention for the stand alone integration of data in accordance with a common schema. Modifying Bowman-Amuah to separate data access from the completion of business functions would require a change in principle in the operation of the Bowman-Amuah invention. As a result, the teachings of the cited combination of documents are not sufficient to render the claims prima facie obvious.
3. Change in the basis for integration. Ranger teaches the integration of data using data reliability and rules on a one at a time basis as required to support a query (see pages 21 and 23, Evidence Appendix, Ranger C2, L20 - 40, C19, L32 – C20, L43). The Examiner has proposed using Ranger in combination with Bowman Amuah to among other things render obvious an invention for integrating data in accordance with a common schema. Modifying Ranger to integrate data in accordance with a common schema instead of rules and reliability would require a change in principle in the operation of the Ranger invention. As a result, the teachings of the cited combination of documents are not sufficient to render the claims prima facie obvious.

The fifth reason that claims 34 - 43 are patentable is that the claimed invention produces results that are concrete, tangible and useful. In view of the previously documented shortcomings in the cited combination of documents that were used as the basis of the claim rejection, it also clear that the claims describe an invention that is novel, surprising, new and non-obvious. Furthermore, the claimed invention produces results that help satisfy a long felt need for enhanced capabilities for data management and for analyzing, managing and optimizing the financial performance of a value chain.

Issue 2 - Whether claims 44 - 52 are patentable under 35 USC 103 over Bowman-Amuah in view of Ranger?

The claims are patentable in view of the shortcomings in the arguments used to support the rejection of the claims and the usefulness of the results produced by the claimed invention. In particular, claims 44 - 52 are allowable for the first, second, third, fourth and fifth reasons advanced under Issue 1.

Issue 3 - Whether claims 62 – 64, 68 – 70, 90, 91 and 134 are patentable under 35 USC 103 over Bowman-Amuah in view of Ranger?

The claims are patentable in view of the shortcomings in the arguments used to support the rejection of the claims and the usefulness of the results produced by the claimed invention. In particular, claims 62 – 64, 68 – 70, 90, 91 and 134 are allowable for the first, second, third, fourth and fifth reasons advanced under Issue 1.

Conclusion

For the extensive reasons advanced above, Appellant respectfully but forcefully contends that each claim is patentable. Therefore, reversal of all rejections is courteously solicited.

Respectfully submitted,

B.J. Bennett, President
Asset Reliance, Inc.
Dated: October 22, 2006

CLAIMS APPENDIX

34. A computer readable medium having sequences of instructions stored therein, which when executed causes a processor in a computer to perform a data preparation method, comprising :integrating data from a variety of systems using xml and a common schema to support organization processing.

35. The computer readable medium of claim 34 where the common schema includes an organization designation.

36. The computer readable medium of claim 35 wherein the designated organization is a single product, a group of products, a division, a company, a multi-company corporation or a value chain.

37. The computer readable medium of claim 34 where the common schema includes a data structure.

38. The computer readable medium of claim 37 where the data structure is a hierarchy.

39. The computer readable medium of claim 34 where the common schema includes a data dictionary.

40. The computer readable medium of claim 39 where the data dictionary defines standard data attributes from the group consisting of account numbers, components of value, currencies, elements of value, units of measure and time periods.

41. The computer readable medium of claim 34 where data are obtained from the group consisting of advanced financial systems, basic financial systems, alliance management systems, brand management systems, customer relationship management systems, channel management systems, intellectual property management systems, process management systems, vendor management systems, operation management systems, sales management systems, human resource systems, accounts receivable systems, accounts payable systems, capital asset systems, inventory systems, invoicing systems, payroll systems, purchasing systems and combinations thereof.

42. The computer readable medium of claim 34 wherein at least a portion of the data are from the Internet or an external database.

43. The computer readable medium of claim 34 where the data preparation method further comprises converting data to match a common schema and storing the converted data in a central database.

44. A data preparation method, comprising:

integrating data from a variety of systems using xml and a common schema to support organization processing.

45. The method of claim 44 where the common schema includes an organization designation and data structure.

46. The method of claim 45 wherein the designated organization is a single product, a group of products, a division, a company, a multi-company corporation or a value chain.

47. The method of claim 44 where the common schema includes a data dictionary

48. The method of claim 47 where the data dictionary defines standard data attributes from the group consisting of account numbers, components of value, currencies, elements of value, units of measure and time periods.

49. The method of claim 44 where data are obtained from the group consisting of advanced financial systems, basic financial systems, alliance management systems, brand management systems, customer relationship management systems, channel management systems, intellectual property management systems, process management systems, vendor management systems, operation management systems, sales management systems, human resource systems, accounts receivable systems, accounts payable systems, capital asset systems, inventory systems, invoicing systems, payroll systems and purchasing systems.

50. The method of claim 44 wherein at least a portion of the data are from the Internet or external databases.

51. The method of claim 44 where the data preparation method further comprises converting and storing data in accordance with a common schema.

52. A computer readable medium having sequences of instructions stored therein, which when executed cause the processors in a plurality of computers connected via a network to perform the data preparation method of claim 44.

53 – 61 (cancelled without prejudice)

62. A computer readable medium having sequences of instructions stored therein, which when executed cause a set of processors in a plurality of computers that have been connected via a network to perform an organization management method, comprising:

integrating a plurality of organization related data from a variety of sources in accordance with a common schema,

using at least a portion of said data to create one or more tools for organization management, and

making the one or more tools available for review

where the one or more tools for organization management further comprise a system for automated trading of organization equity and tools selected from the group consisting of analytical models, category of value models, component of value models, market value models, network models, optimization models, simulation models, value chain models, management reports, lists of changes that will optimize one or more aspects of organization financial performance and combinations thereof.

63. The computer readable medium of claim 62 where the one or more tools are made available for review using an electronic display, a paper document or combinations thereof.

64. The computer readable medium of claim 62 where data are obtained from advanced financial systems, basic financial systems, alliance management systems, brand management systems, customer relationship management systems, channel management systems, estimating systems, intellectual property management systems, process management systems, supply chain management systems, vendor management systems, operation management systems, enterprise resource planning systems (ERP), material requirement planning systems

(MRP), quality control systems, sales management systems, human resource systems, accounts receivable systems, accounts payable systems, capital asset systems, inventory systems, invoicing systems, payroll systems, purchasing systems, web site systems, the Internet, external databases, user input and combinations thereof.

65 – 67 (cancelled without prejudice).

68. The computer readable medium of claim 62, where the common schema defines common attributes selected from the group consisting of data structure, organization designation, data dictionary and combinations thereof.

69. The computer readable medium of claim 68 where the data dictionary defines standard data attributes from the group consisting of account numbers, components of value, currencies, elements of value, organization designations, time periods and units of measure.

70. The computer readable medium of claim 68 where the data structure is a hierarchy.

71 - 89. (cancelled without prejudice)

90. The computer readable medium of claim 62, wherein the one or more aspects of organization financial performance are selected from the group consisting of organization revenue, organization expense, organization capital change, organization current operation value, organization real option value, organization market sentiment value, organization market value and combinations thereof.

91. The computer readable medium of claim 62, wherein the identified changes are changes to alliance value drivers, brand value drivers, channel value drivers, customer value drivers, customer relationship value drivers, employee value drivers, equipment value drivers, intellectual property value drivers, partnership value drivers, process value drivers, production equipment value drivers, vendor value drivers, vendor relationship value drivers, organization equity and combinations thereof.

92 – 133 (cancelled without prejudice)

EVIDENCE APPENDIX

Pages 20 - 23	excerpt from reference first cited May 3, 2006
Pages 24 - 30	excerpt from reference first cited May 23, 2003

US006301584B1

(12) **United States Patent**
Ranger

(10) **Patent No.: US 6,301,584 B1**
(45) **Date of Patent: Oct. 9, 2001**

(54) **SYSTEM AND METHOD FOR RETRIEVING ENTITIES AND INTEGRATING DATA**

(75) Inventor: **Denis Ranger**, Morris Plains, NJ (US)

(73) Assignee: **Home Information Services, Inc.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/137,937**

(22) Filed: **Aug. 21, 1998**

Related U.S. Application Data

(63) Continuation-in-part of application No. 08/915,662, filed on Aug. 21, 1997, now Pat. No. 5,999,940.
(60) Provisional application No. 60/056,523, filed on Aug. 21, 1997.

(30) **Foreign Application Priority Data**

May 28, 1998 (EP) 98201847

(51) **Int. Cl.**[7] **G06F 17/30**
(52) **U.S. Cl.** **707/103**; 707/505; 345/329
(58) **Field of Search** 705/26; 706/54; 345/349, 329; 395/683; 707/201, 100, 103, 102, 505; 709/206

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,491,820		2/1996	Belove et al.	707/3
5,560,005		9/1996	Hoover et al.	707/10
5,644,764	*	7/1997	Johnson et al.	707/103
5,659,736	*	8/1997	Hasegawa et al.	707/100
5,717,925	*	2/1998	Harper et al	707/102
5,740,549		4/1998	Reilly et al.	705/14
5,761,500		6/1998	Gallant et al.	707/10
5,761,663		6/1998	Largarde et al.	707/10
5,809,502		9/1998	Burrows	707/7
5,893,913	*	4/1999	Brodsky et al.	707/201
5,895,470	*	4/1999	Pirolli et al.	707/102
5,948,058	*	9/1999	Kudoh et al.	709/206
5,999,179	*	12/1999	Kekic et al.	345/349
5,999,940	*	12/1999	Ranger	707/103
6,014,637	*	1/2000	Fell et al.	705/26
6,016,393	*	1/2000	White et al.	395/683
6,081,798	*	6/2000	Johnson et al.	706/54
6,166,732	*	12/2000	Mitchell et al.	345/329
6,169,993	*	1/2001	Shutt et al.	707/103
6,192,381	*	2/2001	Stiegemeier et al.	707/505

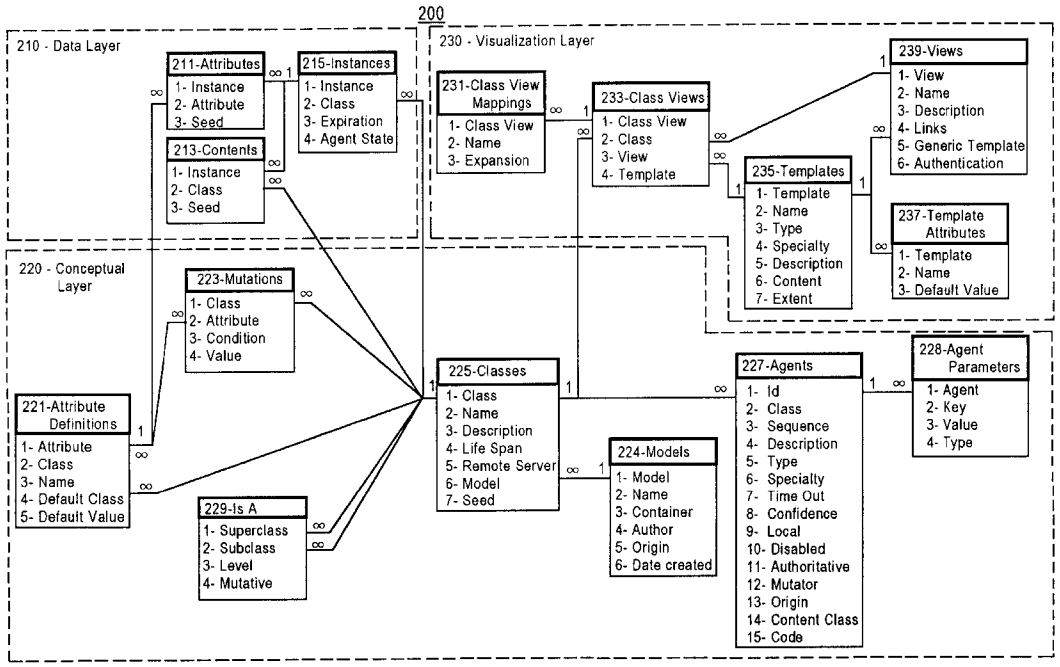
* cited by examiner

Primary Examiner—Thomas Black
Assistant Examiner—Charles L. Rones
(74) *Attorney, Agent, or Firm*—McDermott, Will & Emery

(57) **ABSTRACT**

A data integration system and method gathers information dynamically from one or more data sources, which may be located at different servers and have incompatible formats, structures the information into a configurable, object-oriented information model, and outputs the information for the user according to an associated, configurable visual representation with automatic content classification.

27 Claims, 12 Drawing Sheets



SYSTEM AND METHOD FOR RETRIEVING ENTITIES AND INTEGRATING DATA

RELATED APPLICATIONS

This application is a continuation-in-part of U.S. patent application Ser. No. 08/915,662, filed Aug. 21, 1997, now U.S. Pat. No. 5,999,940 entitled "Interactive Discovery Tool and Methodology," issued on Dec. 7, 1999 by Denis Ranger, the contents of which are incorporated by reference herein, and claims the benefit of U.S. Provisional Application No. 60/056,523, entitled "Method of Data Integration," filed on Aug. 21, 1997 by Denis Ranger, the contents of which are incorporated by reference herein.

FIELD OF THE INVENTION

The present invention relates to data processing and, more particularly, to information discovery and visualization.

BACKGROUND OF THE INVENTION

There is a vast amount of information in the world today that is available by computer. For example, on the World Wide Web alone there are millions of web pages. In addition to the Internet, companies have set up local "intranets" for storing and accessing data for running their organizations. However, the sheer amount of available information is posing increasingly more difficult challenges to conventional approaches.

A major difficulty to overcome is that information relevant to a purpose of a user is often dispersed across the network at many sites. It is often time-consuming for a user to visit all these sites. One conventional approach is a search engine. A search engine is actually a set of programs accessible at a network site within a network, for example a local area network (LAN) at a company or the Internet and World Wide Web. One program, called a "robot" or "spider," pre-traverses a network in search of documents and builds large index files of keywords found in the documents.

A user of the search engine formulates a query comprising one or more keywords and submits the query to another program of the search engine. In response, the search engine inspects its own index files and displays a list of documents that match the search query, typically as hyperlinks. When a user activates one of the hyperlinks to see the information contained in the document, the user exits the site of the search engine and terminates the search process.

Search engines, however, have their drawbacks. For example, a conventional search engine suffers from obsolescence of data in its search indexes due to pre-traversing a network to index documents. Documents are constantly being updated, but it may take months for the new information to filter down to search engines. Furthermore, a search engine is oriented to discovering textual information only. In particular, conventional search engines are not well-suited to indexing information contained in structured databases, e.g. relational databases, and mixing data from incompatible data sources is difficult in conventional search engines.

Attempts have been made to present search results in an object-oriented fashion by homogenizing the search results into an "entity" that is an instance of a specified class, which may be hierarchically dependent upon another "base" class. A class specifies the attributes or properties of an entity, and a dependent class includes the attributes of the base class and additional attributes. A problem with such attempts is that the particular data returned for a particular entity is restricted

to the attributes defined for the specified class of the entity. This restriction means that if the entity to be returned actually belongs to a dependent class, hierarchically dependent upon the specified class, the number of attributes returned to the user will be limited to the properties for the base class, not the dependent class. Consequently, some search results will be not be found and presented to the user. If, however, the user wants to check if a particular entity belongs to a dependent class, another query to the system has to be submitted, specifying the particular dependent class. This checking operation becomes more time consuming as more dependent classes are specified and more entities are found.

SUMMARY OF THE INVENTION

There exists a need for a mechanism to collect relevant information located at a plurality of sites and stored in plurality of incompatible formats according to configurable search strategies.

These and other needs are met by the present invention, which dynamically gathers information from a diversity of data sources with agents, organizes the information in an configurable, information model, and visualizes the information according to a view.

Accordingly, one aspect of the invention relates to an entity retrieving system connectable to at least one data source comprising a memory and a processor connected to an interface. The memory stores a number of classes, in which each class defines the structure of an entity, including property definitions that identify property values stored in the data sources and to be retrieved dedicated to the property definition. The classes include at least one dependent class that is hierarchically linked to at least one other class and contains additional property definitions specifying additional property values, in addition to the property values of the class from which it depends.

The processor, in cooperation with the interface, is configured for receiving a query, which includes an identifier for identifying a particular class and at least one of the property values. The processor also selects, among the classes, the particular class dedicated to the identifier under control of said query, accesses the data sources, retrieves property values pertaining to at least one particular entity that comprises that property value, and outputs the retrieve entities. Upon establishing that the particular entity pertains to one of said dependent classes of the selected particular class, the processor is configured to retrieve the additional properties of the dependent class. According to another aspect, the processor is configured for invoking a plurality of agents concurrently to gather the requested information from the data sources.

Additional objects, advantages, and novel features of the present invention will be set forth in part in the description that follows, and in part, will become apparent upon examination or may be learned by practice of the invention. The objects and advantages of the invention may be realized and obtained by means of the instrumentalities and combinations particularly pointed out in the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. **1** is a high-level block diagram of a computer system with which an embodiment of the present invention can be implemented.

payroll purposes. When an "employee" instance is resolved, the actual class of the instance is one of the two subclass, "exempt" or "nonexempt."

On the other hand, if such an instance is not cached in the data layer **210**, then the instance is instantiated in step **504** with attributes initialized from the seed parameter and the default values in the attribute description, e.g. in the **231-5** field. Instantiation results in the creation of a new entry in the "Instances" table **215** with a unique instance identifier being stored in the "Instance" field **215-1**. In addition, the "Agent Seed" field **215-5** is initialized to the seed parameter and the "Agent State" field **215-4** is cleared.

In step **506**, a "puzzle" is set up that determines which agents are to be invoked for gathering information for the new instance. These agents may be agents specified for the class identified by the class parameter ("class agents") and non-local agents of superclasses of the class ("non-local superclass agents"). In one embodiment, agents are listed in respective entries of the "Agents" table **227**. Class agents are determined from entries in which the class identifier in the "Class" field **227-2** matches the class parameter received in step **500**. Non-local superclass agents are determined from entries in which the "Local" field **227-9** is false and the class identifier in the "Class" field **227-2** matches the class identifier specified in the "Superclass" field **229-1** of the "Is A" table **229** wherein the corresponding "Subclass" field **229-2** contains the class identifier matching the input class parameter.

As described in more detail hereinafter, the puzzle is run, invoking agent to gather data and then integrating the data into one or more entities (step **508**). If successful, the one or more entities are cached in the data layer **210** (step **510**), setting the "Expiration" field **215-3**, as appropriate. For example, the "Expiration" field **215-3** may contain the termination date of a mortal object (cf. the "Life Span" field **225-4**). When a mortal object has expired, it is removed from the data layer **210**. Finally, the instance identifier and the actual class, possibly changed due to a mutation, of the instance is returned in step **512**.

Since agents are invoked when an instance is resolved, information that is potentially more up-to-date can be retrieved than through conventional search engines. Conventional search engines pre-traverse the web to build their index files, which may become out of date for months until the search index is re-updated. With the present invention, however, the "Life Span" attribute controls how long any information object is cached, reducing the obsolescence of information stored at the server to individually acceptable levels, e.g. caching for only a month.

Invoking Agents

Referring to FIG. **6**, running a puzzle results in invoking agents to dynamically access, collect, and integrate "pieces" of data from data sources. More specifically, the agents associated with the class (and superclasses) of the entity to be retrieved are examined. In step **600**, queries are built as a combination of an agent and a "pieece" of information as an input parameter, typically a previously determined attribute for the entity to be retrieved such as a seed value. For example, an agent may get additional information about a person based on a social security number. Given the social security number, a query is created in conjunction with the agent, using the social security number as an input parameter.

On systems that support multi-tasking, all the built queries are launched concurrently at step **602**. Launching a query involves invoking (or executing) an agent with the corresponding piece of information as an input value. The result of launching a query is a result code and, if appropriate, a list of pieces. The result codes are REFRESH_AND_CONTINUE, REFRESH_AND_QUIT, FAIL_AND_CONTINUE, and FAIL_AND_QUIT. "REFRESH" means that the query was successful, while "FAIL" means that the query was unsuccessful (e.g. time out or not found in the data source). "CONTINUE" means that the result is incomplete and "QUIT" means that the query result is controlling, whether successful or unsuccessful. A piece is an attribute, value pair, such as "Name='Bob Smith'".

Generally, agents come in two flavors, attribute agents and content agents, specified in the "Type" field **227-5** of the "Agents" table **227**. An attribute agent is responsible for gathering information about an instance itself, for example, getting the author of a document, the size of the document, and creation date. Attribute agents are normally invoked during instance resolution, which takes place the first time the value of an attribute is requested. In the example, the agent that discovered the length of employment for an employee from an authoritative database is an attribute agent.

Content agents are responsible for gathering the content of the object, for example, getting files in a directory, graphics from a web page, or names from a telephone book. Content agents are invoked whenever content of the object is first accessed, usually when producing a visualization for the object's space. In the example, the agent that discovered files in a directory is a content agent.

To support concurrent query execution, queries use a common "blackboard" to post their results. When a query is launched, the blackboard is first checked for an entry listing the agent and piece. If the entity is incomplete, because another query is currently running, then the query waits until the result from the running query is available and returns the result posted on the blackboard. On the other hand, if there is not entry for the agent and piece, then such an entry in the blackboard is created, the agent is invoked, and the results are posted on to the blackboard and returned.

When an agent is invoked, it is passed an instance identifier for accessing and modifying attributes of the instance being resolved and the input seed value. For example, if the instance is a member of a "employee" class and the seed value is an employee number, the agent is passed an identifier of the instance and the employee number. The agent may use the employee number to query an authoritative database (cf. the "Authoritative" field **227-11**), parse the result to determine some values of attributes (such as length of employment), and initialize the attributes with the parsed values. As another example, a "directory" object may use a pathname as a seed value. The contents, e.g. files and other directories, of a directory having that pathname may be inspected by the agent for creating file objects as contents of the directory object.

At step **604**, the results of launching the queries are processed as they come in. If the query failed to run due to a timeout condition (e.g. with a result code of FAIL_AND_CONTINUE), then the query is placed on a failed queries list. If the query has failed and the agent is considered to be authoritative (result code of FAIL_AND_QUIT), then all remaining agents are marked as done and the search for this puzzle is terminated. If the query has failed, but not due to a time-out (also FAIL_AND_CONTINUE), then the agent is simply marked as done, but the other, concurrently

invoked agents are allowed to continue. Results of a content query are added to the content of the current result. Attribute queries, on the other hand, add their results to the attributes of the current result. Failed queries are retried in step **606**.

In the example illustrated in FIG. **9**, an agent dedicated to the Product class, is provided for retrieving the Supplier and Type property values based on the ID number. These property values are for example stored in an internal data source, for example a relational database **246**. The agent comprises an address in field Origin **227-13** indicating the path name of the database **246** data source. In order to enable to retrieve data from different types of data sources, there are provided different types of agents. For a relational database such as Oracle ®, the agent is an ODBC agent type. The agent further comprises a series of instructions indicating which data from the addressed data source are to be retrieved by the agent, for example:

"SELECT Key, Type, Supplier FROM Products"

The agent further comprises in its agent parameters **228** for assigning, for each property value to be retrieved, a portion of the data to one of the property definitions. In this case, "Key" is assigned to "ID" property definition, "Type" to "Type" property definition and "Supplier" to "Supplier" property definition.

This agent co-operates with interface **111** for accessing the data source, under control of processor **104** and for retrieving the requested data. In the example mentioned hereinabove, the following data will be returned: "93-21123" forming the ID, "Doubleday" forming the Supplier and "Book" forming the type.

Data Integration

When several agents retrieve, from different data sources, property values that should correspond, some property values retrieve might not be equal to each other. For example, a customer's telephone number may be recorded differently in two data sources, or there might be three different authors for the same book title. In the first case, it is probable that the same customer has two phone numbers (an inconsistency), in the second case, we may be dealing with three altogether different books (an ambiguity).

Inconsistencies and ambiguities are virtually unavoidable when integrating multiple data sources that were not conceived together and that may not even be managed by the same organization. There is therefore a need for appropriately handling ambiguities and inconsistencies within data. The manner in which an embodiment handles these problems is explained by means of an example.

Assume that agents are looking for a Person named Bob Smith. Agent A is configured to look for a person's address given the person's name. Agents B and C are configured to look for a person' age given the person's name, each agent targeting a separate data source. This example is illustrated in FIG. **10**.

Agent A returns with not one but two "Bob Smith", one living in New York and the other in Newark. Determining whether there are two persons named Bob Smith or only one with a conflicting address depends on how much to trust Agent A to be accurate or, in other words, whether its data source contains the correct addresses. For this purpose, a reliability or confidence parameter **227-8** is assigned to the agent. If the confidence parameter for agent A is 100%, then there are two persons named Bob Smith and two entities are thus shown to the user. On the other hand, if Agent A has a confidence parameter of only 10%, then the one entity is produced, showing two possibilities for a property value, e.g. "New York OR Newark".

Assume now agent A has a 100% reliability parameter. Agent B and C for the Bob Smith in New York obtain his age. Both agree that it is 35. However agents B and C for the Bob Smith in Newark disagree about his age. Agent B indicates 24 and Agent C 27. In this case, Agents B and C are fallible, but their disagreement is not sufficient grounds to see two separate persons named Bob Smith living in Newark. If Agents B and C have substantially the same reliability parameter that is relatively low, for example 10%, then one entity will be presented to the user with an indication of two property values for the age: "24 OR 27", such as illustrated in FIG. **10**. In this situation, there is a "conflict of opinion" between data sources about the age the Bob Smith living in Newark. Because of ambiguities and inconsistencies, a request to an embodiment to find an entity may end up returning more than one entity, with some "conflicts of opinion" about some of them. When this occurs, the user is presented with a display using the generic template **239-5** for the requested view, e.g. a Web page, that gives a choice between these entities and highlights conflicts.

If Agents B and C have substantially the same reliability parameter, which is relatively high, for example 90%, then on embodiment interprets that there are two distinct entities as being two separate entities which will be presented to the user, each with its own age. If agent B is substantially more reliable than agent C, for example agent B is at least 25% more reliable than agent C, then an embodiment will prefer the property value retrieved by agent B, i.e. 24, and only the entity retrieved having this value will be presented to the user.

Consequently, providing a reliability parameter for agents, inconsistencies and ambiguities in property values can be interpreted, filtering out unreliable property values or presenting them in an appropriate fashion to the user.

When it is determined that two or more entities are to be created, for example two persons named Bob Smith, instances are created for each new entity. For each new entity, a new corresponding sub-puzzle is set up and then run. At this point, the top-level puzzle switches to a passive mode in which the top-level puzzle waits for all the sub-puzzles to finish and return their results recursively.

Mutations

Sometimes, information discovered for an entity, typically by an attribute agent, causes the entity to change its class. Accordingly, the entity is checked if a mutation should be performed to change the class of the entity (step **608**). In a particular check, mutation patterns or mutation agents dedicated to one of the dependent classes of the current entity are checked. This checking can be performed by verifying, for each dependent class, if the mutative field **229-4** is true. If true, then mutation patterns or mutation agents dedicated to the classes "Book" and "Audio Tape" are examined. A mutation pattern dedicated to the classes book comprises a condition, for example: "If the Product Type ="book" then mutate the Product into a Book", which is evaluated to determine if the found property value for the product type falls within the condition. For this purpose, the processor **102** compares the property values stored in the memory **104** with the condition of the mutation pattern. In the example, the retrieved property for the product type is a "Book". Thus, a mutation occurs and the class of the entity becomes "Book" causing additional property values pertaining to the class "Book" to be retrieved. A mutation agent is a stored procedure or other piece of procedural logic that can

US006332163B1

(12) **United States Patent**
Bowman-Amuah

(10) **Patent No.: US 6,332,163 B1**
(45) **Date of Patent: Dec. 18, 2001**

(54) **METHOD FOR PROVIDING COMMUNICATION SERVICES OVER A COMPUTER NETWORK SYSTEM**

(75) Inventor: **Michel K. Bowman-Amuah**, Colorado Springs, CO (US)

(73) Assignee: **Accenture, LLP**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/387,642**

(22) Filed: **Sep. 1, 1999**

(51) **Int. Cl.**[7] **G06F 13/00**
(52) **U.S. Cl.** **709/231**; 709/217; 709/223; 709/227; 709/329
(58) **Field of Search** 709/102, 202, 709/203, 217, 218, 219, 223, 225, 227, 230, 231, 238, 329

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,301,320		4/1994	McAttee et al.	395/650
5,457,797	*	10/1995	Butterworth et al.	709/302
5,721,908		2/1998	Lagarde et al.	395/610
5,764,955	*	6/1998	Doolan	709/223
5,867,153	*	2/1999	Grandcolas et al.	345/326
5,890,133		3/1999	Ernst	705/7
5,892,909	*	4/1999	Grasso et al.	709/201
5,907,704		5/1999	Gudmundson et al.	395/701
5,933,816	*	8/1999	Zeannah et al.	705/35
5,940,075	*	8/1999	Mutschler, III et al.	345/335
5,953,707		9/1999	Huang et al.	705/10
6,041,365	*	3/2000	Kleinerman	709/302

FOREIGN PATENT DOCUMENTS

WO 99/08208	2/1999	(WO)	G06F/17/30

OTHER PUBLICATIONS

Microsoft Corporation, *Microsoft Solutions Framework Overview A Quick Tour of the MSF Models*, URL: http://channels. microsoft.com/enterprise/support/support/consult, Viewed Oct. 9, 1999.

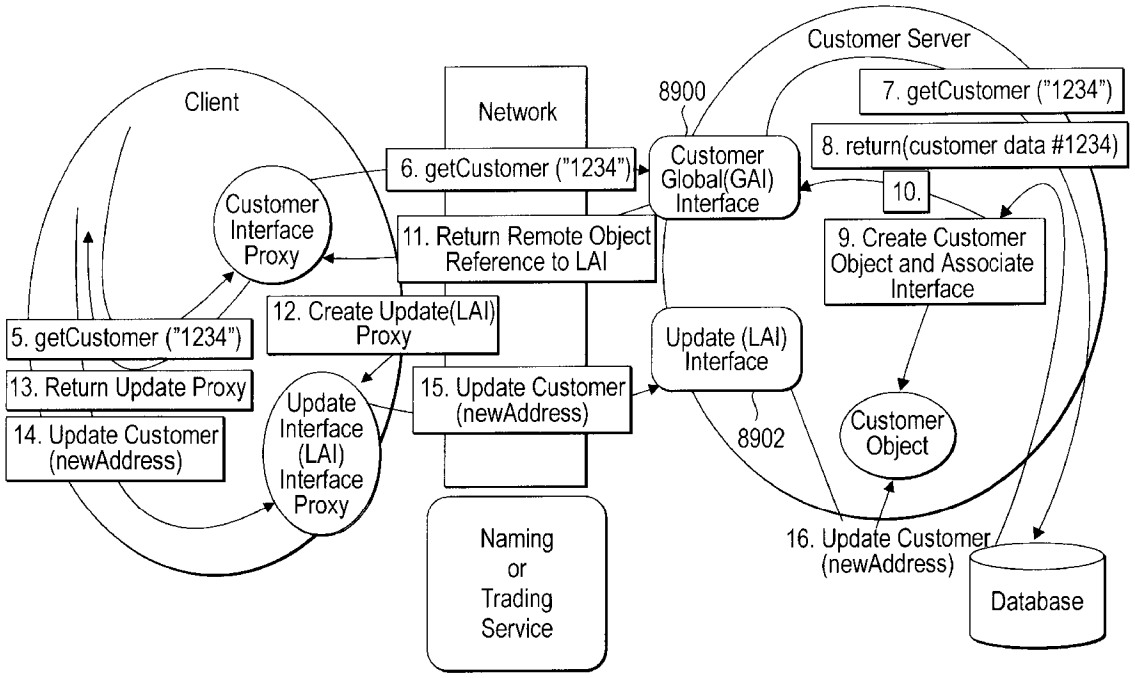
* cited by examiner

Primary Examiner—Viet D. Vu
(74) *Attorney, Agent, or Firm*—Oppenheimer Wolff & Donnelly, LLP; Stefanie M. Howell

(57) **ABSTRACT**

A system, method and article of manufacture are provided for implementing communication services patterns. A fixed format stream-based communication system is provided and service is delivered via a globally addressable interface. Access is afforded to a legacy system. Service is delivered via a locally addressable interface. A null value is communicated and data is transmitted from a server to a client via pages. A naming service and a client are interfaced with the naming service allowing access to a plurality of different sets of services from a plurality of globally addressable interfaces. A stream-based communication system is provided and data is efficiently retrieved.

15 Claims, 123 Drawing Sheets



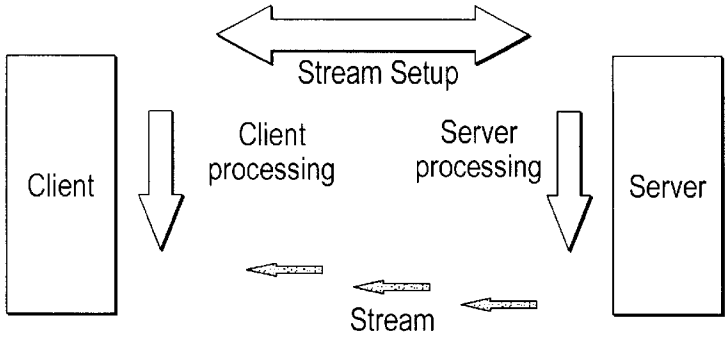


Fig. 20

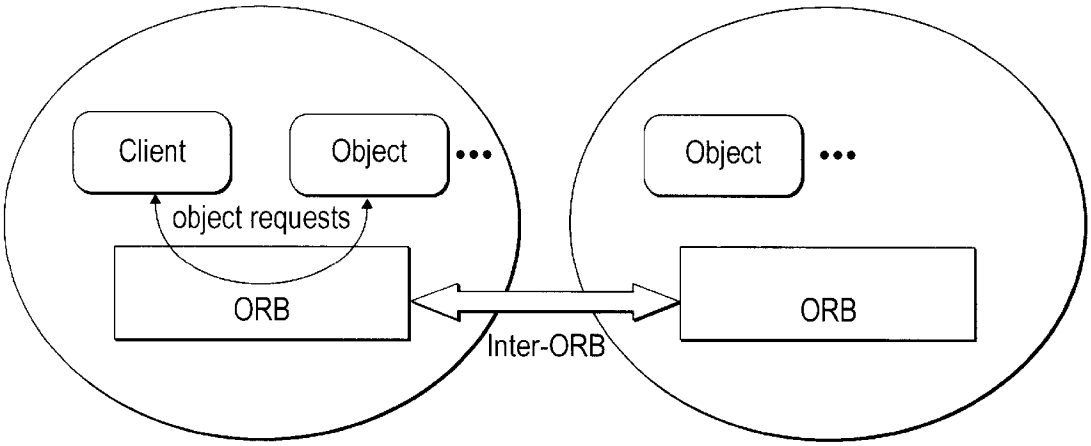


Fig. 21

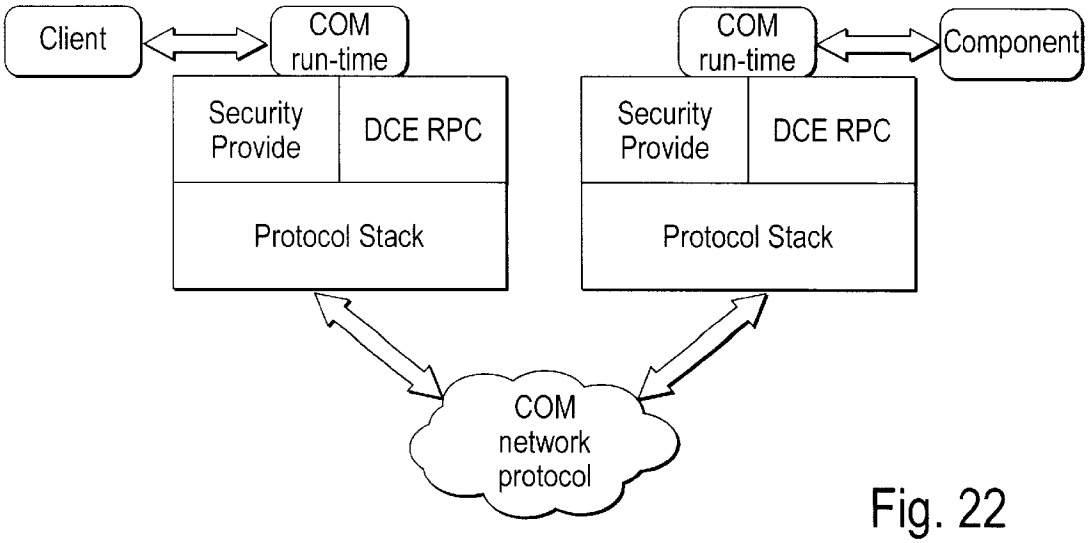


Fig. 22

FIG. **16** illustrates File Sharing services;

FIG. **17** depicts Message Passing services;

FIG. **18** depicts Message Queuing services;

FIG. **19** illustrates Publish and Subscribe services;

FIG. **20** depicts Streaming, in which a real-time data stream is transferred;

FIG. **21** illustrates CORBA-based Object Messaging;

FIG. **22** illustrates COM Messaging;

FIG. **23** represents CTI Messaging;

FIG. **24** illustrates various components of the Communication Fabric of the present invention;

FIG. **25** illustrates the two categories of the Physical Media;

FIG. **26** illustrates several of the components of the Transaction areas of the Netcentric Architecture Framework;

FIG. **27** illustrates various components of the Environmental Services of the Netcentric Architecture Framework;

FIG. **28** illustrates the Base Services of the Netcentric Architecture Framework;

FIG. **29** shows the major components of the reporting application framework;

FIG. **30** illustrates an example of how a custom report architecture relates to a workstation platform technology architecture;

FIG. **31** describes the relationships between the major components of the report process and the report writer process;

FIG. **32** shows the module hierarchy for the custom report process;

FIG. **33** depicts the various components of the Business Logic portion of the Netcentric Architecture Framework;

FIG. **34** illustrates a relationship between major themes that impact aspects of software development and management;

FIG. **35** illustrates how components are viewed from different perspectives;

FIG. **36** shows a relationship between business components and partitioned business components;

FIG. **37** shows how a Billing Business Component may create an invoice;

FIG. **38** illustrates the relationship between the spectrum of Business Components and the types of Partitioned Business Components;

FIG. **39** illustrates the flow of workflow, dialog flow, and/or user interface designs to a User Interface Component;

FIG. **40** is a diagram of an Application Model which illustrates how the different types of Partitioned Business Components might interact with each other;

FIG. **41** illustrates what makes up a Partitioned Business Component;

FIG. **42** illustrates the role of patterns and frameworks;

FIG. **43** illustrates this Business Component Identifying Methodology including both Planning and Delivering stages;

FIG. **44** shows a high level picture of application component interaction for an Order Entry system;

FIG. **45** illustrates a traditional organization structure including an activities component, a credit/collections component, a billing component, and a finance component;

FIG. **46** provides an illustration of a horizontal organization model;

FIG. **47** illustrates a workcell organization approach including an activities component, a credit/collections component, a billing component, and a finance component;

FIG. **48** illustrates the Enterprise Information Architecture (EIA) model;

FIG. **49** illustrates a V-model of Verification, Validation, and Testing;

FIG. **50** portrays of a development architecture with a seamless integration of tools which can be plugged in for the capture and communication of particular deliverables;

FIG. **51** shows a design architecture with the compromises made for today's component construction environment;

FIG. **52** illustrates a business process to object mapping;

FIG. **53** is a diagram which illustrates a graph of resilience to change;

FIG. **54** illustrates a flowchart for a method for providing an abstraction factory pattern in accordance with an embodiment of the present invention;

FIG. **55** illustrates a flowchart for a method for representing a plurality of batch jobs of a system each with a unique class in accordance with an embodiment of the present invention;

FIG. **56** illustrates a class diagram of the batch job hierarchy;

FIG. **57** illustrates an object interaction graph of a possible implementation of the class diagram of FIG. **56**;

FIG. **58** illustrates a flowchart for a method for controlling access to data of a business object via an attribute dictionary in accordance with an embodiment of the present invention;

FIG. **59** illustrates a flowchart for a method for structuring batch activities for simplified reconfiguration in accordance with an embodiment of the present invention;

FIG. **60** illustrates the manner in which the AttributeDictionaryClient is the facade which delegates to the AttributeDictionary;

FIG. **61** depicts the use of the containsKey() method on the HashMap to ensure that the value will exist before the get() method is used;

FIG. **62** illustrates a method that dictates that any nullPointerException that is thrown would be caught and rethrown as the more user-friendly exception in the attribute dictionary pattern environment;

FIG. **63** illustrates the Get the Attribute Names method in the attribute dictionary pattern environment;

FIG. **64** illustrates a flowchart for a method for managing constants in a computer program in accordance with an embodiment of the present invention;

FIG. **65** illustrates a flowchart for a method for providing a fixed format stream-based communication system in accordance with an embodiment of the present invention;

FIG. **66** illustrates two systems communicating via a stream-based communication and using a common generic format to relay the meta-data information;

FIG. **67** illustrates an example of a Fixed Format message associated with the fixed format stream patterns;

FIG. **68** depicts the complete Fixed Format Stream pattern associated with the fixed format stream patterns;

FIG. **69** illustrates fixed format contracts containing meta-data information for translating structured data onto and off of a stream;

FIG. **70** illustrates a Customer object in an object-based system streaming itself into a stream, the stream being sent

ing key data-window relationship. This allows the user to work in a controlled and, well managed, environment.

Web Browser **1308**

Web Browser Services allow users to view and interact with applications and documents made up of varying data types, such as text, graphics, and audio. These services also provide support for navigation within and across documents no matter where they are located, through the use of links embedded into the document content. Web Browser Services retain the link connection, i.e., document physical location, and mask the complexities of that connection from the user. Web Browser services can be further subdivided into: Browser Extension, Form, and User Navigation.

Parlez-vous Internet?

The Elements of Web Style

Language philosopher Benjamin Whorf once said, "We dissect nature along lines laid down by our native language. Language is not simply a reporting device for experience, but a defining framework for it." This notion is especially true when applied to the World Wide Web. The evolution of the Web from a rigid, text-centric village to an elastic, multimedia-rich universe has been driven by modifications to the languages behind it. The Internet is at a crucial point in its development as a number of enhancements for extending Web technology come under scrutiny by Internet standards groups. These enhancements will ultimately push the Web into the realms of distributed document processing and interactive multimedia.

SGML: in the beginning . . .

Although the World Wide Web was not created until the early 1990s, the language behind it dates back to the genesis of the Internet in the 1960s. Scientists at IBM were working on a Generalized Markup Language (GML) for describing, formatting, and sharing electronic documents. Markup refers to the practice in traditional publishing of annotating manuscripts with layout instructions for the typesetters.

In 1986, the International Standards Organization (ISO) adopted a version of that early GML called Standard Generalized Markup Language (SGML). SGML is a large and highly-sophisticated system for tagging documents to ensure that their appearance will remain the same regardless of the type of platform used to view them. Designers use SGML to create Document Type Definitions (DTDs), which detail how tags (also known as format codes) are defined and interpreted within specified documents. These tags can be used to control the positioning and formatting of a document's text and images. SGML is used for large, complex, and highly-structured documents that are subject to frequent revisions, such as dictionaries, indexes, computer manuals, and corporate telephone directories.

HTML: SGML for dummies?

While creating the World Wide Web in the early 1990s, scientists at CERN discovered that in spite of its power and versatility, SGML's sophistication did not allow for quick and easy Web publishing. As a result, they developed HyperText Markup Language (HTML), a relatively simple application of SGML. This simplicity has contributed to the exponential growth of the Web over the last few years. HTML files are written in plain text and can be created using any text editor from the most robust Web page authoring software (such as Microsoft's FrontPage or Sausage Software's HotDog) to the anemic Notepad utility included with Microsoft's Windows operating system.

As with many languages, HTML is in a state of constant evolution. The World Wide Web Consortium W3C oversees new extensions of HTML developed by both software companies (such as Microsoft and Netscape Communications) and individual Web page authors and ensures that each new specification is fully-compatible with previous ones. Basic features supported by HTML include headings, lists, paragraphs, tables, electronic forms, in-line images (images next to text), and hypertext links. Enhancements to the original HTML 1.0 specification include banners, the applet tag to support Java, image maps, and text flow around images.

The W3C also approved the specification for version 4.0 of HTML (http://www.w3.org/TR/REC-html40). This specification builds upon earlier iterations of HTML by enabling Web authors to include advanced forms, in-line frames, and enhanced tables in Web pages. HTML 4.0 also allows authors to publish pages in any language, and to better manage differences in language, text direction, and character encoding.

Perhaps most significantly, HTML 4.0 increases authors' control over how pages are organized by adding support for Cascading Style Sheets CSS Style sheets contain directions for how and where layout elements such as margins, fonts, headers, and links are displayed in Web pages. With CSS, authors can use programming scripts and objects to apply multiple style sheets to Web pages to create dynamic content. CSS can also be used to centralize control of layout attributes for multiple pages within a Web site, thus avoiding the tedious process of changing each page individually.

Dynamic HTML: Dyn-o-mite!

HTML's simplicity soon began to limit authors who demanded more advanced multimedia and page design capabilities. Enter Dynamic HTML DHTML As an extension of HTML, DHTML allows Web pages to function more like interactive CD-ROMs by responding to user-generated events. DHTML allows Web page objects to be manipulated after they have been loaded into a browser. This enables users to shun plug-ins and Java applets and avoid bandwidth-consuming return trips to the server. For example, tables can expand or headers can scurry across the page based on a user's mouse movements.

Unfortunately, the tremendous potential offered by DHTML is marred by incompatible standards. At the heart of the DHTML debate is a specification called the Document Object Model DOM The DOM categorizes Web page elements—including text, images, and links—as objects and specifies the attributes that are associated with each object. The DOM makes Web document objects accessible to scripting languages such as JavaScript and VisualBasic Script (VBScript), which can be used to change the appearance, location, and even the content of those objects in real-time.

Microsoft's Internet Explorer 4.0 supports a W3C "Working Draft" DOM specification that uses the CSS standard for layout control and Web document object manipulation. In contrast, Netscape's implementation of DHTML in Communicator 4.0 uses a proprietary "Dynamic Layers" tag, which assigns multiple layers to a page within which objects are manipulated. As a result, Web pages authored using either version of DHTML may not be viewed properly using the other's browser. XML: X marks the spot

HTML 4.0 and Dynamic HTML have given Web authors more control over the ways in which a Web page is displayed. But they have done little to address a growing problem in the developer community: how to access and manage data in Web documents so as to gain more control over document structure. To this end, leading Internet developers devised Extensible Markup Language (XML), a watered-down version of SGML that reduces its complexity

while maintaining its flexibility. Like SGML, XML is a meta-language that allows authors to create their own customized tags to identify different types of data on their Web pages. In addition to improving document structure, these tags will make it possible to more effectively index and search for information in databases and on the Web.

XML documents consist of two parts. The first is the document itself, which contains XML tags for identifying data elements and resembles an HTML document. The second part is a DTD that defines the document structure by explaining what the tags mean and how they should be interpreted. In order to view XML documents, Web browsers and search engines will need special XML processors called "parsers." Currently, Microsoft's Internet Explorer 4.0 contains two XML parsers: a high-performance parser written in C++ and another one written in Java.

A number of vendors plan to use XML as the underlying language for new Web standards and applications. Microsoft uses XML for its Channel Definition Format, a Web-based "push" content delivery system included in Internet Explorer 4.0. Netscape will use XML in its Meta Content Framework to describe and store metadata, or collections of information, in forthcoming versions of Communicator. XML is currently playing an important role the realm of electronic commerce via the Open Financial Exchange, an application developed by Microsoft, Intuit, and CheckFree for conducting electronic financial transactions. Similarly, HL7, a healthcare information systems standards organization, is using XML to support electronic data interchange EDI of clinical, financial, and administrative information (http://www.mcis.duke.edu/standards/HL7/sigs/sgml/index.html).

Meet cousin VRML

In 1994, a number of Internet thought leaders, including Tim Berners-Lee—the "father" of the Web—met to determine how they could bring the hot, new technology known as virtual reality VR to the Web. VR refers to the use of computers to create artificial and navigable 3-D worlds where users can create and manipulate virtual objects in real time. This led to the creation of Virtual Reality Modeling Language (VRML—pronounced "ver-mul"). VRML is technically not a markup language because it uses graphical rather than text-based file formats.

In order to create 3-D worlds and objects with VRML, users need a VRML editor such as Silicon Graphics' Cosmo Worlds (http://cosmo.sgi.com/products/studio/worlds). To view VRML content, users need either a VRML browser or a VRML plug-in for standard HTML browsers. Leading VRML plug-ins include Cosmo Player from Silicon Graphics (http://vrml.sgi.com/cosmoplayer), Liquid Reality from Microsoft's DimensionX subsidiary (http://www.microsoft.com/dimensionx), OZ Virtual from OZ Interactive (http://www.oz.com/ov/main_bot.html), and WorldView from Intervista (http://www.intervista.com/products/worldview/index.html), These plug-ins can typically be downloaded for free from the Web.

VRML is capable of displaying static and animated objects and supports hyperlinks to multimedia formats such as audio clips, video files, and graphical images. As users maneuver through VRML worlds, the landscape shifts to match their movements and give the impression that they are moving through real space. The new VRML 2.0 specification finalized in August 1996 intensifies the immersive experience of VR worlds on the Web by enabling users to interact both with each other and with their surroundings. Other new features supported by VRML 2.0 include richer geometry description, background textures, sound and video, multilingual text, Java applets, and scripting using VBScript and JavaScript. VRML will become a significant technology in creating next-generation Internet application as the language continues to mature and its availability increases.

The future: give us a big SMIL

The Web has come a long way since the codification of HTML 1.0. It has moved from simple text-based documents that included headings, bulleted lists, and hyperlinks to dynamic pages that support rich graphic images and virtual reality. So what next for the Web? The answer resides in a Synchronized Multimedia Integration Language (SMIL), a new markup language being developed by the W3C. SMIL will allow Web authors to deliver television-like content over the Web using less bandwidth and a simple text editor, rather than intricate scripting.

SMIL is based on XML and does not represent a specific media format. Instead, SMIL defines the tags that link different media types together. The language enables Web authors to sort multimedia content into separate audio, video, text, and image files and streams which are sent to a user's browser. The SMIL tags then specify the "schedule" for displaying those components by determining whether they should be played together or sequentially. This enables elaborate multimedia presentations to be created out of smaller, less bandwidth-consuming components.

Implementation Considerations

Many features such as graphics, frames, etc. supported by Web Browsers today were not available in initial releases. Furthermore, with every new release the functionality supported by Web Browsers keeps growing at a remarkable pace.

Much of the appeal of Web Browsers is the ability to provide a universal client that will offer users a consistent and familiar user interface from which many types of applications can be executed and many types of documents can be viewed, on many types of operating systems and machines, as well as independent of where these applications and documents reside.

Web Browsers employ standard protocols such as Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP) to provide seamless access to documents across machine and network boundaries.

The distinction between the desktop and the Web Browser narrowed with the release of Microsoft IE 4.0, which integrated Web browsing into the desktop, and gave a user the ability to view directories as though they were Web pages. Web Browser, as a distinct entity, may even fade away with time.

Exemplary products that may be used to implement this component includes Netscape Navigator; Netscape Communicator; Microsoft Internet Explorer; Netscape LiveWire; Netscape LiveWire Pro; Symantec Visual Cafe; Microsoft Front Page; Microsoft Visual J++; IBM VisualAge.

Execution Products:

- Netscape Navigator or Communicator—one of the original Web Browsers, Navigator currently has the largest market share of the installed browser market and strong developer support. Communicator is the newest version with add-on collaborative functionality
- Microsoft Internet Explorer (IE)—a Web Browser that is tightly integrated with Windows and supports the major features of the Netscape Navigator as well as Microsoft's own ActiveX technologies.

Development Products:

Web Browsers require new or at least revised development tools for working with new languages and standards such as HTML, ActiveX and Java. Many browser content

call-level SQL variants and supersets. Depending upon the underlying storage model, non-SQL access methods may be used instead.

Many of the Netcentric applications are broadcast-type applications, designed to market products and/or publish policies and procedures. Furthermore, there is now a growth of Netcentric applications that are transaction-type applications used to process a customers sales order, maintenance request, etc. Typically these type of applications require integration with a database manager. Database Services include: Storage Services, Indexing Services, Security Services, Access Services, and Replication/Synchronization Services

Implementation Considerations

The core database services such as Security, Storage and Access are provided by all major RDBMS products, whereas the additional services of Synchronization and Replication are available only in specific products.

Product Considerations

Oracle 7.3; Sybase SQL Server; Informix; IBM DB/2; Microsoft SQL Server

Oracle 7.3—market leader in the Unix client/server RDBMS market, Oracle is available for a wide variety of hardware platforms including MPP machines. Oracles market position and breadth of platform support has made it the RDBMS of choice for variety of financial, accounting, human resources, and manufacturing application software packages. Informix—second in RDBMS market share after Oracle, Informix is often selected for its ability to support both large centralized databases and distributed environments with a single RDBMS product. Sybase SQL Server—third in RDBMS market share, Sybase traditionally focused upon medium-sized databases and distributed environments; it has strong architecture support for database replication and distributed transaction processing across remote sites.

IBM DB2—the leader in MVS mainframe database management, IBM DB2 family of relational database products are designed to offer open, industrial strength database management for decision support, transaction processing and line of business applications. The DB2 family now spans not only IBM platforms like personal computers, AS/400 systems, RISC System/6000 hardware and IBM mainframe computers, but also non-IBM machines such as Hewlett-Packard and Sun Microsystems. Microsoft SQL Server—the latest version of a high-performance client/server relational database management system. Building on version 6.0, SQL Server 6.5 introduces key new features such as transparent distributed transactions, simplified administration, OLE-based programming interfaces, improved support for industry standards and Internet integration.

Replication/Synchronization **1404**

Replication Services support an environment in which multiple copies of databases must be maintained. For example, if ad hoc reporting queries or data warehousing applications can work with a replica of the transaction database, these resource intensive applications will not interfere with mission critical transaction processing. Replication can be either complete or partial. During complete replication all records are copied from one destination to another, while during partial replication, only a subset of data is copied, as specified by the user or the program. Replication can also be done either real-time or on-demand (i.e., initiated by a user, program or a scheduler). The following might be possible if databases are replicated on alternate server(s): better availability or recoverability of distributed applications; better performance and reduced network cost, particularly in environments where users are widely geographically dispersed; etc.

Synchronization Services perform the transactions required to make one or more information sources that are intended to mirror each other consistent. This function may especially valuable when implementing applications for users of mobile devices because it allows a working copy of data or documents to be available locally without a constant network attachment. The emergence of applications that allow teams to collaborate and share knowledge has heightened the need for Synchronization Services in the execution architecture.

The terms Replication and Synchronization are used interchangeably, depending on the vendor, article, book, etc. For example, when Lotus Notes refers to Replication, it means both a combination of Replication and Synchronization Services described above. When Sybase refers to Replication it only means copying data from one source to another.

Implementation Consideration

Replication/Synchronization Services are sometimes supplied as part of commercial databases, document management systems or groupware products such as Lotus Notes, Microsoft Exchange, Oracle, etc.

With Windows 95 and Windows NT 4.0, Microsoft has also introduced the concept of Replication/Synchronization Services into the operating system. Through the briefcase application users can automatically synchronize files and SQL data between their Windows PC and a Windows NT server. Underlying this application is the user-extensible Win32 synchronization services API which can be used to build custom synchronization tools.

Are changes in data usage anticipated?

Data can be dynamically changed to accommodate changes in how the data is used.

Is it desirable to shield the user from the data access process?

A replicated database often consolidates data from heterogeneous data sources, thus shielding the user from the processes required to locate, access and query the data.

What are the availability requirements of the system?

Replication provides high availability. If the master database is down, users can still access the local copy of the database.

Is there a business need to reduce communication costs?

Depending on the configuration (real time vs. nightly replication, etc.), there is a potential to reduce communications costs since the data access is local.

Is scalability an issue?

With users, data, and queries spread across multiple computers, scalability is less of a problem.

Can users benefit from the increased performance of local data access?

Access to replicated data is fast since data is stored locally and users do not have to remotely access the master database. This is especially true for image and document data which cannot be quickly accessed from a central site. Making automatic copies of a database reduces locking conflicts and gives multiple sets of users better performance than if they shared the same database.

Product Considerations

What is the current or proposed environment?

Platforms supported as well as source and target DBMS should be considered.

enables the data access method to be changed at runtime (e.g. batch mode, online mode or Request Batcher).

The Stream-Based Communication pattern can be used to stream the business object's data to the handler. The stream can then be either forwarded to a Request Batcher or can parsed and sent to database.

Individual Persistence

FIG. **163** illustrates a flowchart for a method **16300** for organizing data access among a plurality of business entities. Data about a user is retrieved and packaged into a cross-functional data object in operation **16302** and **16304**. A request for data is retrieved from one of a plurality of business objects in operation **16306**. In operation **16308**, the business object are directed to the data object such that the business object retrieves the entire data object. The business object also selects the data from the data object.

Both locking and integrity may use a uniform mechanism. The business object may retrieve account, customer, and bill-related data from the data object. Also, the business objects may be able to update themselves independently of each other.

Optionally, new business objects may take advantage of existing data access routines. Also, each business object may use a uniform access architecture.

Create a data access architecture that supports reusable, independent business objects in the context of atomic, functionally-specific transactions.

A business unit of work, or business transaction, typically acts on multiple business entities. But for each individual entity, the transaction might only display and update a few data attributes.

For example, accepting bill payment over the phone might use the account number, customer name, amount due, date due, and credit card number. The transaction could therefore avoid accessing many attributes of the account, customer, or monthly bill entities. This might naturally lead to a data architecture which only fetches required attributes, based on the transaction's needs.

Indeed, a traditional client/server program retrieves data in a piecemeal fashion. In this case, the example program would typically allocate a single record to fetch and store the required data items. Then, an "accept bill payment" data access module would fill this structure. This couples data access to processing function.

FIG. **164** illustrates retrieving data piecemeal.

This traditional style of data access may seem flexible. After all, developers can grab whatever data they need for a particular business transaction.

But such access is very unstructured. Different pieces of a cohesive account entity, for example, scatter across multiple windows. Some windows will use the account's effective date; others will not.

This also introduces redundancy. Retrieving the date requires determining the correct database, table, column, and type declaration. Yet another developer who needs this date, for a different data set, duplicates the effort. This does not encapsulate changes, thereby raising costs for testing, maintenance, and extension.

Moreover, each transaction must hand-craft its own retrieval procedure. Creating the thousandth new business transaction would require creating a thousandth new access module. Yet all data items would already have been retrieved by other modules. This style of data access is not reusable.

Finally, business entities are typically less likely to change than the transactions, or processes, which act on those entities. For example, an enterprise might re-engineer its billing function. Regardless of the resulting process, the account, customer, and monthly bill objects would likely remain. This suggests that transaction-based data access is brittle.

Therefore, data access should be organized around business entities, rather than transactions. Individual Persistence packages data into cross-functional objects, rather than transaction-specific data structures. Each individual business object, instead of the window or application, manages the retrieval of its data items.

A business object provides public methods for accessing, comparing, displaying, and setting that data. Developers can therefore no longer plunder the persistent store, selecting data items at will. Instead, they must access their data through encapsulated, self-requesting business objects.

With this architecture, the example billing function retrieves account, customer, monthly bill, and bill payment objects.

FIG. **165** illustrates the manner in which the present invention retrieves whole objects **16500**.

For reuse, business objects should be able to request and update themselves independently of each other. Separating the data access for customer and account objects supports reusing them in isolation. Objects should therefore avoid explicitly requesting other linked objects, unless a formal containment relationship exists.

Finally, separation of concern allows business objects to remain blissfully unaware of the transactions which use them. A business object will not know which data items or services it may need to provide to its clients. Thus, the object must bring back all its data.

Benefits

- Reuse. New transactions can take advantage of existing data access routines. For example, introducing a new business transaction, like perform credit check, would use existing customer and account objects. Yet, these domain model objects would already know how to update themselves. Therefore, the new application would build no new data access code.
- Maintainability and extensibility. This approach supports "fix in one place." Any changes to particular data elements only need to be changed, tested, and recompiled in one access module, that of the owning business object.
- Uniformity. Both optimistic locking and referential integrity (RI) are typically defined at the business object level. For example, separate account and customer objects typically have their own locking attributes. In addition, an RI rule usually relates one entity to another, such as "all accounts must have a customer." Organizing data access around business entities simplifies locking and integrity. Both can use a uniform mechanism, implying that the architecture can hide technical details. This avoids the hard-coding typical of the transaction-based approach.
- Flexibility. Whole object retrieval is extensible. It allows a transaction to ask an object for any data. This supports maintenance and extension. A developer can easily change the particular data items a transaction uses. But whole retrieval still guarantees that those items will already have been retrieved. For example, a future release of the accept bill payment window could also display the social security number. Yet the data access routines would need no modification.

Related Proceedings Appendix

None